DISTRIBUTED DATA ACQUISITION FOR BNL802 II: SOFTWARE

W. A. Watson III and M. J. LeVine
Physics Department, Brookhaven National Laboratory
Upton, New York, 11973

CONF-870552--11
BNL--39807
DE87 010446

Abstract

Multiple processes running on a VAX and on VME-based processors allow up to 16 detector sub-systems to run independently or coupled, with an aggregate throughput of 600 Kbytes/sec. VMS facilities are used extensively for command definition, message passing, controlling access to CAMAC and high voltage modules, and maintaining shared data structures.

Introduction

The Distributed Data Acquisition (DDA) System was developed at Brookhaven National Laboratory for experiment 802, part of the relativistic heavy ion program at BNL. The detector hardware includes a charged particle spectrometer with four tracking chambers, a time-of-flight wall, and Čerenkov counters for particle identification. In addition, there is a charged particle multiplicity array, a lead glass wall, and a zero degree calorimeter. The necessity of dealing with several thousand channels of ADC and TDC information from these detectors led to the use of multiple processors for parallel readout of the event data. The purpose of this paper is to describe the higher level software for this distributed system. The acquisition hardware and lower level software is described in another paper, *Distributed Data Acquisition for BNL802 I: The Front End*[1].

The DDA system consists of a VAX 11/785 host and a VME-based intelligent front end (see Figure 1). The front end is managed by a Motorola 68010 microprocessor (called the Chairman) running a multi-tasking operating system. Each CAMAC crate is interfaced to a dedicated 68000 CPU [2], so that all CAMAC crates may be read out in parallel. The software for each crate is written in assembly language and is tailored to the detector components attached to that crate. Fastbus crates are read out by LeCroy 1821 segment managers, through a DMA interface to VME developed for this system [3]. A second VME crate linked by a DMA crate interconnect contains 15 Motorola 68020 processor elements with 2 Mbytes of memory each. These are used to compress and format the data, and eventually for making third level trigger decisions. When the system is not in use for data acquisition, these processors will be used as a data analysis engine.

The front end is connected to the VAX host by a DR11-W link using a message passing scheme for data and control information. Currently there are two high speed tape drives, a gigabyte of disk space, and 16 Mbytes of memory on the host, with plans to add laser disks in the near future.

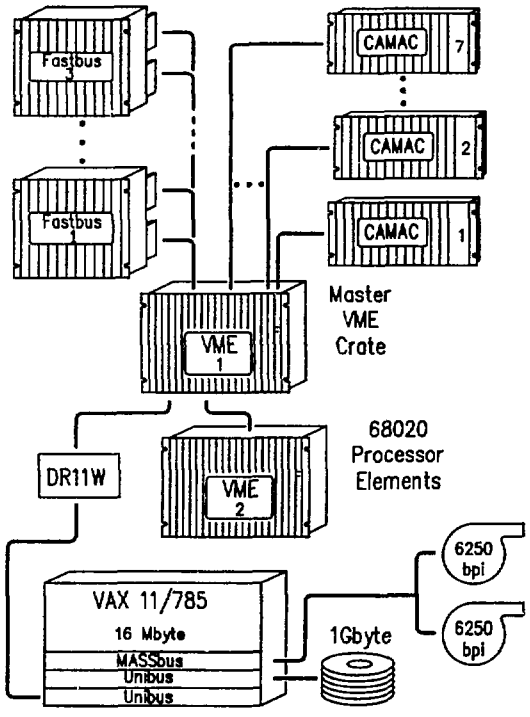

Figure 1: DDA Hardware

Front End Software

The Chairman's software currently consists of 5 main tasks (see Figure 2). These are currently written in Fortran, and are downloaded from the host over the DR11-W link in less than half a minute.

Event Stream

In the first stage of the data pipeline, the Event Builder task collects event fragments from the CAMAC interfaces and from the Fastbus task. This process is interrupt driven and involves no interaction with the operating system. Each detector sub-system (referred to as a partition) may be operated either independently or coupled with one or more of the other partitions. Events from uncoupled partitions are maintained in separate data streams which can be started, stopped, and logged independently.

In the second stage of the event pipeline, full 16 Kbyte buffers from the Event Builder are queued to

the Format and Filter task. This task passes buffers through a DMA crate interconnect to 68020 processor elements for further formatting and possibly a third-level trigger decision. Processors are allocated dynamically, and events are kept in chronological order. The formatting of the data includes translating geographical address tags in the data stream into logical, or partition, address tags. The translation tables are maintained on the host, and downloaded over the link. Formatted events from the processor elements are repacked into buffers and queued for transmission to the host by the VME Link Server task.

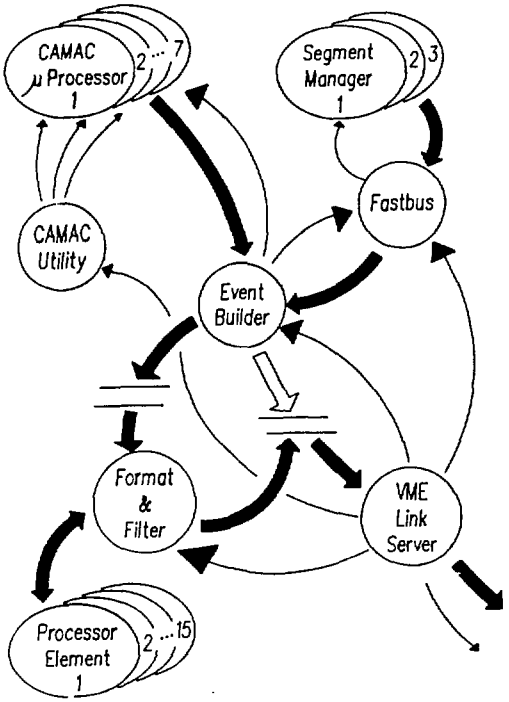

Figure 2: Front End Tasks

At high data rates, the Event Builder could fill buffers faster than they can be processed by the formatting task, which runs at a lower priority. The spill structure of the accelerator allows the slower task to catch up at the end of the spill period. In this way, complete formatting of the event data does not slow down data acquisition. In the same way, buffer transmission to the VAX is asynchronous with data acquisition.

Control Messages

Control of the front end is by command messages passed from the VAX host over the DR11-W link. Each message is tagged with a message type, a number from 0 to 511. The message type determines which task is to receive the message, and the VME Link Server forwards the message to the appropriate task. The overhead to do this is extremely low, on the order of 200 μsec including the context switch.

Collisions on the VAX–VME link can in principle occur and are handled by re-transmission. Priority is given to the VAX host so that control messages are not delayed by data transmission. In practice, few collisions occur since most messages are data from VME to the host.

Also shown in the figure is a separate CAMAC control task for utility CNAF's and high voltage control. This task is not part of the event stream, and is mainly used for monitoring purposes.

Host Software

The host software consists of a large number of programs, most of which are activated in response to a user command. Three processes are always active for communication with the front end: the Link Server, the Buffer Logger, and the Buffer Processor.

Real-Time Processes

On the host, data buffers are received by the VAX Link Server (Figure 3). If logging is enabled for the coupled or uncoupled stream to which the buffer belongs, it is queued to the Buffer Logger and immediately written to a log file. Both the Link Server process and the Buffer Logger process on the host run at elevated priorities, allowing data acquisition and logging to proceed at the speed of the link, approximately 600 Kbytes/sec. This is well matched to the capabilities of the high speed tape drive, filling a tape in 4 minutes. The overhead for transferring data from VME to tape is approximately 4% of the CPU per 100 Kbytes/sec. Device independent output routines are used, allowing the output to be re-directed to disk, or to laser disks in the future.

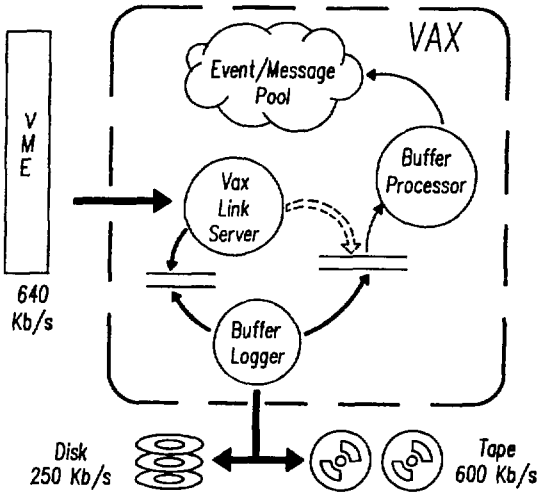

Figure 3: Real-Time Processes

After logging, a sample of the events is copied into an event pool by the Buffer Processor for distribution to one or more analysis programs. Software to implement the event pool was developed by Quarrie for CDF at Fermilab [4]. The Buffer Processor samples the input stream with an overhead of 10% per 50 events/sec + 5% per 100 Kbytes/sec. It runs at normal priority, and therefore it does not interfere with the ability of the Buffer Logger to keep up with the data coming from the front end.

Control of the Front End

Control of the front end is through messages constructed by a host program, for example in response to a user command to change a high voltage. The control message is queued to the VAX Link Server, transmitted across the link, and dispatched by the VME Link Server to the appropriate task. The reply, if any, is automatically routed back to the originating VAX process through the event pool (Figure 4). This is accomplished by dynamically allocating a unique message number for the reply, and sending it with the message. For reply messages like this which do not contain event data, the Buffer Processor delivers all messages instead of sampling them.

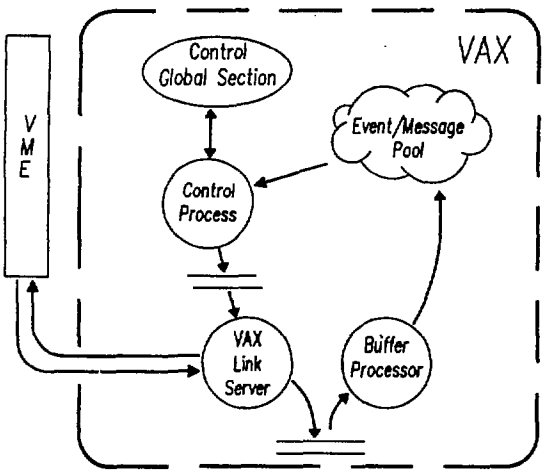

Figure 4: Control Flow

Some operations, such as CAMAC CNAF commands and commands to alter high voltages, are protected in that they require that the user belong to the set of people allowed to operate on the corresponding detector. This is implemented through the use of VMS identifiers to define the user groups, and by defining domains (sets of CAMAC slots or high voltage outputs) upon which only holders of the identifiers may operate. Users may belong to more than one partition, and it is also possible to assign a single slot to more than one domain.

The definitions of the domains are kept in a memory resident structure called a global section. This global section also contains other heavily used information, and serves as a type of high speed database for control operations. Items in this database are quickly accessed by an 8 character name through hashing techniques, and can be defined and deleted dynamically.

In order to reduce the load on the link during a spill, and also to further reduce the probability of collisions, many control utilities synchronize their operations to the spill structure of the accelerator. These programs, such as the high voltage monitor and the scaler display program, wait for a message from the front end indicating that a spill has finished. They then issue their read requests and receive their replies during the interval between spills. If there is no spill structure available, they can also operate in a time based mode.

Data Analysis

Online monitoring and analysis programs receive events from the event pool (Figure 5). The event pool software allows multiple "consumers", and allows events to be modified and written back to the pool by "producers". In this way, analysis pipelines may be constructed online.

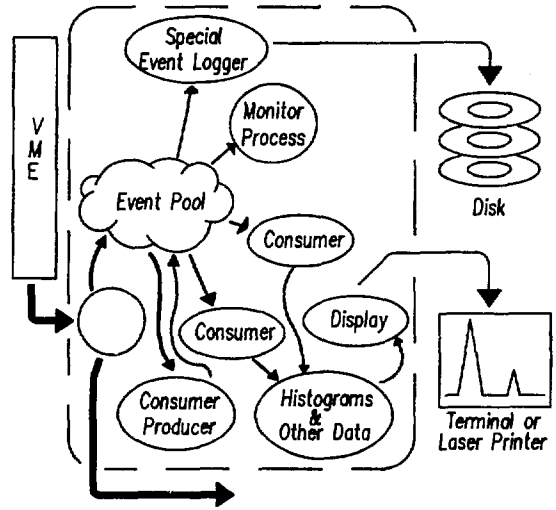

Figure 5: Data Analysis

Analysis programs run as subprocesses, and place histograms, list data (N-tuples), and other output into analysis global sections. These analysis sections may be examined by display programs, thereby allowing simultaneous histogramming and display. Furthermore, since the histograms are not "inside" the analysis program, they can be manipulated interactively (integrated, etc.) at any point, and can be recovered after a program crash or even after a machine crash by periodically flushing the modified pages to disk. Currently 1D, 2D, and list data (N-tuples) are supported using byte, word, long-

word, and real datatypes. Histograms are accessed by 8 character name or by integer number using the same memory management routines as are used in the control global section.

Summary information, in the form of calibration constants and the like, are stored on disk using Rdb, DEC's relational database. This database is also used to store translation tables, Fastbus ADC pedestals, and for many other bookkeeping chores. While Rdb may not have all the features of Oracle or Ingres, it is an extremely useful tool, and its comparitively low price makes it an excellent choice for VAX systems.

User Interface

The user interacts with DDA through commands entered at the DCL prompt, using the command definition utilities to extend the DCL language. In this way, the full capabilities of VMS are always available, including command files with DCL and DDA commands interspersed. The syntax is already familiar to most VAX users, and to assist in learning DDA, there is a standard VMS help library available for all DDA commands. There are currently 36 DDA verbs, most of which have arguments and qualifiers.

Line mode commands within many of the DDA utilities, such as the interactive graphics program and the high voltage utility, also use a DCL-like syntax. The graphics program includes its own help library and the ability to spawn any DCL or DDA command.

As previously mentioned, data analysis is done in subprocesses, leaving the user free to enter additional DDA or DCL commands. Communication with these subprocesses can take one of two forms. In the simplest form, a DDA command is used to send a one line mail message to the subprocess. If more extended interaction is necessary, the parent process is suspended, and an interactive session with the subprocess may be used for arbitrary tasks like program initialization or option selection. When the interaction is finished, the parent process is re-activated.

Summary

VMS has proven to be an extremely useful toolbox for the development of a data acquisition system. Global sections can be dynamically created to share data among many programs, or from one invocation of a program to the next. The VMS lock manager has been used to control access to the directory structures used in the analysis and control global sections. VMS identifiers have proven useful in grouping individuals into overlapping classes of users. The ability to expand the DCL language and help facilities provides an easily extensible and easy to learn system.

On the hardware side, the large number of commercially available and inexpensive VME components has made it possible to build a powerful front end at a very low cost. We anticipate being able to easily upgrade this system in the future by purchasing faster CPU's and faster memories as they become available.

Acknowledgment

The authors gratefully acknowledge the contributions of Betty McBreen in the later phases of this project.

This work was supported by the U. S. Department of Energy, Division of Basic Energy Sciences under Contract No. DE-AC02-76CH00016.

References

[1] M. J. LeVine, W. A. Watson III, H. von der Schmitt, and S. Kaufman, "Distributed Data Acquisition for BNL802 I: The Front End."

[2] R. A. Scheetz and M. J. LeVine, "An Intelligent VME-based CAMAC Crate Controller."

[3] R. A. Scheetz and M. J. LeVine, "A VME-VMX Interface to Fastbus via the LeCroy 1821 Segment Manager."

[4] D. R. Quarrie, "The CDF Data Acquisition System," IEEE Trans. Nucl. Sci., NS32-4 1397.